A2

that a software release is bug free is by testing the software with a wide variety of test cases, carefully chosen to test all critical modules in the software product. Such testing procedures are especially important where the software under test is designed to operate across a number of processors or other systems operating in parallel. In such cases, the individuals writing the test cases should be familiar with the operating system, and communication protocols for all of the elements in the system, but unfortunately, there are few individuals with all of the skills necessary to write a complete test program.

On page 1, please replace lines 29 and 31, as follows:

A3

The development and execution of software test cases also takes a large investment of time and resources, and can delay the timely introduction of a software product to the marketplace. Because software applications are written and sold in a

On page 2, please replace lines 12-22, as follows:

A4

What is needed is an improved system and method for developing test cases. The present invention satisfies that need by offering a system that relieves the system test designer from writing the automated code from scratch. Instead, the test designer is provided with a library of test objects, each implementing a portion of the automated test procedure. When one of the objects in the library is selected, the user is prompted to select from test object options that define required test parameters, thus simplifying the process. The system and method for developing test cases also relieves the test case designer from the burden of familiarizing themselves with the protocols and interoperability requirement for each and every system element used by the software, and allows the test plan to be updated and documented with significantly less effort.

On page 4, please replace lines 9-10, as follows:

A5

FIGs. 9A-9E are diagrams showing a listing of an automated test code generated from the script file shown in FIG. 6;

On page 10, please replace lines 3-18, as follows:

A6

FIG. 7 is a flow chart showing an illustrative example of process steps used to generate the test plan 322 from the source file 318. In some cases, the tags in the source file 318 simply specify a print formatting (e.g. indentation and whether the test following the tag is a header, such as tags 401-450). Other tags, such as tags 602A-616A, are instead associated with the members of the library of executable test code objects 314. The first tag type can be interpreted and printed by a QPRINT function 321. However, the second tag type is not recognized by a QPRINT function 321, and must be identified and handled differently. These tags are interpreted, translated using script macros 325, and conversational language statements for the test plan 322 are generated as shown in blocks 702 and 704. This is accomplished via script macros 325 such as the script macro having representative code instructions presented in FIGs. 30A and 30B. If tags are encountered that are uninterpretable (because, for instance, the user typed in the incorrect tag name, a tag not associated with one of the script macros 325, or an unsupported or out of range tag parameter), an error message (such as "+++") noting this error is placed in the test plan 322 in the appropriate place.

0955804 .050501

On page 11, please replace lines 8-21, as follows:

A7

After the test plan 322 has been approved, the next step is to generate test code 320 from the source file 318. This step is performed by an HPTC EXEC software module 319, which interprets the tags and associated tag parameters in the source file, and generates test code with links to the appropriate member of the library of executable code objects 314. In one embodiment, the generated automated test code 320 uses a screen-scraping technique wherein the commands to the system elements are provided via coded automated keystroke entries which are injected to the appropriate system element. An example of a portion of HPTC EXEC code used to translate a HPSRLM2 test object to test code is presented in FIG. 31. An example of the generated test code 320 translated from the HPSRLM2 test object is shown in FIG. 9E, at 910. An example of the subroutine that the test code line 910 calls is

A7

shown in FIGs. 32A-32C. This code resides in the automated executable subroutine library 327. Responsive messages from the system element are intercepted and used to determine whether the command proceeded normally, or if there was an error.

On page 11, please replace lines 22-29, as follows:

A8

FIGs. 9A-9E are diagrams showing a listing of an automated test code 320 generated from the script file shown in FIG. 6. The listing begins in FIG. 9A with commented code ("leading '/*' characters denote code comments) describing the objectives, and scenario for the test case. The listing continues in FIG. 9B with additional information regarding the test case name, the source file, and other information. The remainder of FIG. 9B, as well as FIGs. 9C and 9D describe variable definitions for scenario variables and called commands. FIG. 9E presents the test code commands.

On page 12, please replace lines 1-15, as follows:

A9

Returning to FIG. 3, once the test code has been generated, it can be executed 326. Using the screen-scraping technique described above, the commands are provided to the system elements 132, and response messages from the system elements 132 are used to determine the test result, and whether the test errors were reported. If no test errors were reported 328, the test results are provided to the user, and the test case is completed. If test errors are reported 328, they may be corrected by altering the test case source filed 318, and regenerating the test plan 322 and the automated test code 320. One of the advantages provided by the present invention is the use of the source file 318 to automatically generate both the test plan 322 and the automated test code 320. When errors are detected in either the test plan 322 or the test code 320 (whether by executing the code or compiling the code), those errors can be corrected by making appropriate changes to the source file 318. Hence, the test code 320 is self-documenting, and there is no need for the user to go through the time-consuming (and often omitted) process of rewriting the test plan 322 to conform with actual test procedures.

On page 13, please replace lines 1-3, as follows:

A10 input 210. The test case GUI 312, as described more fully herein, also permits the user to convert manually entered test case commands into test case objects, and to store them in the executable code test object library 313 for future use.

On page 14, please replace lines 13-16, as follows:

A11 FIG. 15 is a diagram showing an example of how the test case GUI 312 prompts the test designer to enter an author name. An author field 1504 is provided in an "author name" window 1502. The test designer enters their name in the author name field 1502, and selects the "next" button 1506.

On page 17, please replace lines 9-19, as follows:

A12 FIG. 26 is a diagram showing another embodiment of the test case GUI 312. Here, the user has selected the terminals category 2602, showing library member objects 2604 including a "use a live terminal" library member object 2606, and has further selected the "use a live terminal" library member object 2606, thus opening a "Setup a live terminal" window 2606. The "Setup a live terminal window" 2606 presents an ON frame 2612, terminal option frame 2608, a terminal frame 2614 and a port frame 2616. The required input parameters for the live terminal library member object 2606 are highlighted by changing the color of the frame or the test labeling of the frame. If the user enters a value in the combo box in the terminal option frame 2608, that value is saved for the duration of the test case development session. Tooltip box 2610 is applied to the ON frame 2612 to provide the user with context sensitive help.

IN THE DRAWINGS

Proposed Drawing Changes is attached indicating a typographical error as follows. "FIG. 9F" should read –FIG. 9E--.